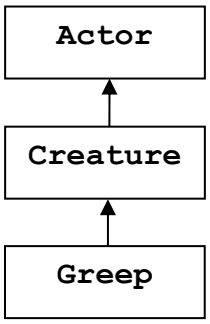
Class Hierarchy



Class `Creature`

(Parent Class of `Greep`)

A `Creature` is the base class for all alien beings in the Greeps scenario. It provides the basic abilities of creatures in this world.

Because a `Greep` is a `Creature`, any of these methods can be utilized in the `Greep` class. The method signatures I believe you will find to be the most helpful to use are listed below.

`atShip`

`public final boolean` **`atShip`**`()`

Returns `true` if we are at our spaceship.

`atWater`

`public boolean` **`atWater`**`()`

Returns `true` if there is water in front of us.

`atWorldEdge`

`public boolean` **`atWorldEdge`**`()`

Returns `true` if we are at the edge of the world.

`carryingTomato`

`public final boolean` **`carryingTomato`**`()`

Returns `true` if we are carrying a tomato.

`dropTomato`

`protected final void` **`dropTomato`**`()`

Drop the tomato we are carrying. If we are at the ship, it is counted. If not, it's just gone...

`getFlag`

`public boolean` **`getFlag`**`(int flagNo)`

Retrieve the value of a flag. '`flagNo`' can be 1 or 2.

---

getMemory

`public int` **`getMemory`**`()`

Retrieve a previously stored value.

---

loadTomato

`public final void` **`loadTomato`**`()`

Load a tomato onto *another* creature. This works only if there is another creature and a tomato pile present, otherwise this method does nothing.

---

randomChance

`protected boolean` **`randomChance`**`(int percent)`

Return `true` in exactly 'percent' number of calls. That is: a call randomChance(25) has a 25% chance to return true.

---

removeTomato

`public final boolean` **`removeTomato`**`()`

Remove the tomato currently carried (and return `true`). Return `false` if we were not carrying one.

---

seePaint

`public boolean` **`seePaint`**`(java.lang.String color)`

Check whether we can see paint of a given color where we are sitting. There are three possible colors: `"red"`, `"orange"`, and `"purple"`.

---

setFlag

```
public void setFlag(int flagNo,
                    boolean value)
```

Store a user-defined `boolean` value (a "flag"). Two flags are available, i.e. `flagNo` may be 1 or 2.

---

setMemory

`public void` **`setMemory`**`(int val)`

Store a user defined value. Attention: even though the parameter type is `int`, only a one-byte sized value (`0 <= val <= 255`) is accepted.

---

spit

`public void` **`spit`**`(java.lang.String color)`

Spit a drop of paint onto the ground. We can spit in three colors: `"red"`, `"orange"`, and `"purple"`. (All other strings will be mapped to one of these.)

---

turn

`public void` **`turn`**`(int angle)`

Turn 'angle' degrees towards the right (clockwise).

---

turnHome

`public void` **`turnHome`**`()`

Turn in the direction facing the home ship.

Class `Actor`

(Parent Class of `Creature`)

`Creature` objects are a subclass of `Actor`. `Creature` objects (and therefore `Greep` objects) inherit the following methods from `Actor` (for more specific information, look at the methods themselves in the `Actor` class). The method signatures I believe you will find to be the most helpful to use are listed below.

`getX`

`public int` **`getX`**`()`
`throws java.lang.IllegalStateException`

Return the x-coordinate of the object's current location. The value returned is the horizontal index of the object's cell in the world.

Returns:

The x-coordinate of the object's current location.

`getY`

`public int` **`getY`**`()`

Return the y-coordinate of the object's current location. The value returned is the vertical index of the object's cell in the world.

Returns:

The y-coordinate of the object's current location

`getRotation`

`public int` **`getRotation`**`()`

Return the current rotation of the object. Rotation is expressed as a degree value, range (0..359). Zero degrees is to the east. The angle increases clockwise.

Returns:

The rotation in degrees.

`setRotation`

`public void` **`setRotation`**`(int rotation)`

Set the rotation of the object. Rotation is expressed as a degree value, range (0..359). Zero degrees is to the east. The angle increases clockwise.

Note that there are some `Actor` methods that are not listed here that are illegal to use for purposes of the competition (as defined in the rules). Specifically they include `setLocation, getNeighbors, getObjectsAtOffset, getOneObjectAtOffset, getObjectsInRange,` among others. Use any of any illegal methods will result in automatic disqualification from the competition. Don't even think about it ☹.